UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS Y SISTEMAS

INTRODUCCIÓN A LA COMPUTACIÓN Y PROGRAMACIÓN 2

SECCIÓN A+

MANUAL DE TÉCNICO

CARLOS OJANI NG VALLADARES

CARNÉ 201801434

MÉTODOS UTILIZADOS

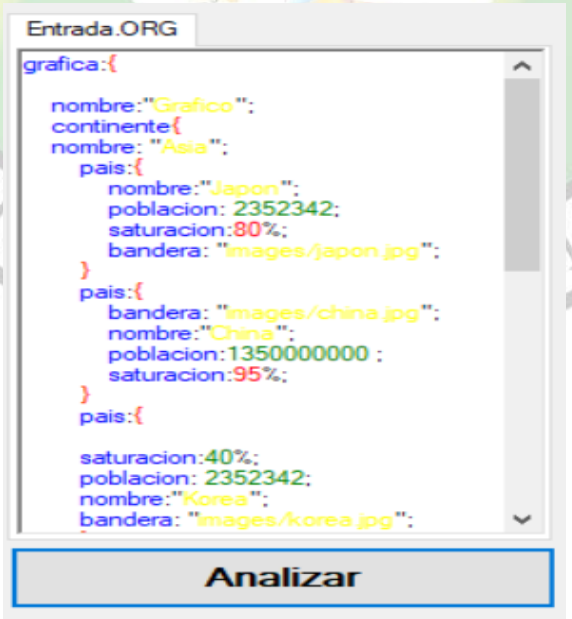
Durante el desarrollo del programa se elaboró una interfaz amigable para el usuario. Se utilizaron dos formularios en total, el primero que cumple con la función de analizar el archivo de entrada y el segundo muestra la tabla de tokens encontrados (Que pueden ser visibles tanto en un archivo html que se genera como en la tabla mencionada).

A simple vista del programa se puede notar lo siguiente:

- Una Tab Control, que puede contener cajas de texto (RichTextBox).
- Una barra de menú con tres opciones que se desglosan.
- Un apartado para mostrar el país seleccionado.
- Un apartado para el gráfico

Primeramente, la barra de menú contiene tres opciones: archivo, ayuda y mostrar tabla.

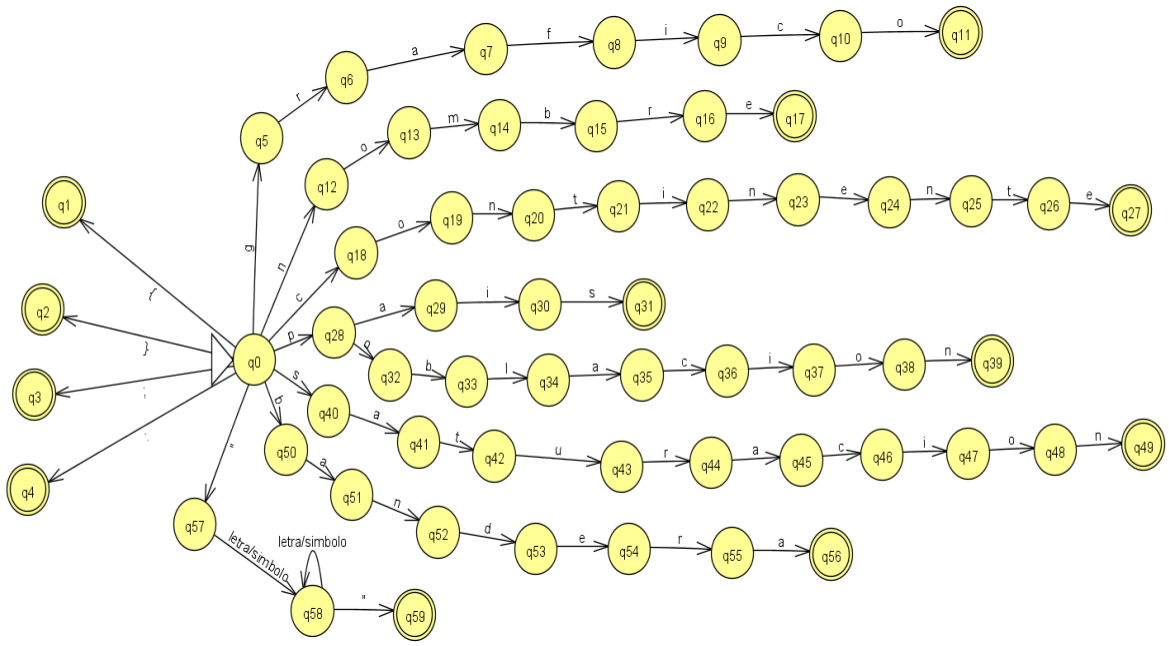
En el manejo del submenú “archivo”, el usuario puede agregar una nueva pestaña al TabControl. El evento que genera “nueva pestaña” crea una nueva TabPage, a su vez, se inicializa un RichTextBox y este mismo se utiliza para llenar la nueva pestaña. La opción de cargar archivo sucede de una forma similar, a diferencia que el usuario carga un archivo .ly de entrada, se lee el texto que está dentro del mismo y se utiliza para rellenar la nueva pestaña que se crea. Algo muy importante para los botones de archivo, es que antes se crea e inicializa un RichTextBox estático que sirve de señuelo o auxiliar. Es de mucha utilidad para el botón guardar, ya que toma en cuenta la caja de texto que está en ese momento en ejecución. Ya que lee línea por línea la caja de texto que está en selección para guardar su contenido. Una vez cargado el archivo correcto, las palabras dentro del archivo cambian de color, dependiendo de su relevancia, como se muestra a continuación:



Para la identificación y organización de tokens, se realizó una clase llamada “Token” el cual contiene una enum llamado Tipo que lista el nombre del tipo de token a identificar. Es decir, si es un numero entero, una llave de cierre, signo mayor, etc… El método constructor de la clase Token recibe tres parámetros: el tipo de token, id del token y valor. A la hora de devolver el tipo del token, este mismos, depende que cual sea, retorna un String con el nombre del token. El valor retorna el símbolo o lexema identificado. Para obtener el id del token, se clasificaron según su tipo, este retorna un número entero. De la misma forma los errores, solo que asignándolos como tipo desconocidos.

Se realizó una clase analizador donde se lleva a cabo toda la lógica correspondiente a la identificación de tokens.

A continuación se muestra el autómata realizado por medio del método del árbol (lógica para realizar el analizador):



En la clase analizador se procede escribir un nuevo método, pero este es una linked list haciendo referencia la clase Token y que tiene como parámetro un String (que es la caja de texto del formulario principal). Ya enviado el String, que es la entrada, se lee carácter por carácter utilizando un ciclo for con limite el largo del texto. Anidado a eso, se encuentra un switch que básicamente lo que hace tomar en que estado se encuentra el carácter a leer (ya que inicialmente se encuentra en 0 que es el inicial), se compara que el carácter empiece tal y como se muestra en el texto dela archivo, si es así, se almacena el carácter en una variable “c” y este mismo se va concatenando al momento de compararse. Si el carácter no es el esperado se tomará como un error léxico y el carácter

guardado en “c” es el ultimo y por consiguiente es el error, se almacena en la lista de errores. Este proceso termina hasta que cada conjunto de caracteres llegue al estado de aceptación, que es cuando termina de concatenarse la palabra. Así mismo con los simbolos que tienen solo dos estados, el inicial y el de aceptación. Las cadenas encerradas en comillas, se leen desde que empieza la primera comilla y se concatena hasta llegar a la segunda comilla, esto consta de tres estados ya que es un ciclo y no es de interés el contenido dentro de las comillas.

En analizador, también existe el método de Agregar Tokens, recibe como parámetro un Tipo de token y cada tipo se agrega a una lista auxiliar de tokens llamada “salida”. Al terminar de agregar cada token regresa al estado actual a 0. Este método es únicamente llamado cuando el conjunto de caracteres llegue al estado de aceptación.

En el formulario principal, se encuentra el botón analizar. El cual utiliza el método escanear de la clase analizador y envía el texto de la RichTextBox. A la hora de terminar de analizar y enlistar los tokens, este mismo recorre la lista de tokens recibida utilizando un foreach leyendo ítem por ítem y comparando. Para generar la lista html se utiliza una lista de tipo String, como un htmles básicamente un archivo de texto, se agrega línea por línea el código html y al final de agregar las líneas se crea un archivo de extensión .html utilizando las líneas escritas. Para generar la tabla de nuevo se recorre la lista con un foreach, a diferencia que a cada línea dentro del ciclo agregamos la estructura html de una tabla (<tr><td></td></tr>) y dentro de cada <td></td> se llena con el ítem que es de tipo Token y se obtiene el valor, tipo, id de token y aparte se crea una variable entera que lleva el conteo del total de la lista. Esto genera la lista html.

Para la creación del gráfico, se utilizó la herramienta graphviz. Básicamente consta de generar una una imagen (.png) e ir concatenando líneas de texto. Para esto se utilizo un método que inicia un proceso dentro del cmd que ejecuta la instrucción del lenguaje dot y genera el archivo de texto que será como se ordena el gráfico y la imagen png. Para construir el texto, se recorrió la lista de tokens y al obtener sus valores, estos se iban concatenando en una lista string. Luego esa lista string se envia como parámetro al método graficar que recibe como parámetro un string, que luego por medio de un stringbuileder junta todo el texto, luego genera la ruta del png y el archivo dot los envia como parámetros al método que ejecuta la instrucción el cmd que genera ambos archivos mencionados. Una vez hecho esto, la imagen se muestra en la picturebox de lado derecho de la ventana. Para llenar los nodos y sus colores, se tomó la saturación de cada país, el valor se convirtió en tipo entero y se fue sumando hasta que se encontrará una llave de cierre de continente (lo mismo para el contador de paises en el continente). Al momento de llegar a la llave de cierre se realiza la operación que promedia la saturación en el contiente: la suma de la saturación dividida el contador de paises que se tiene en el momento. Este resultado se concatena en el texto para generar la gráfica y depende del numero de saturación se compara los valores (convertidos a enteros) con cierto rango de porcentaje. Así, por ejemplo al tener un 80 se compara y se obtiene que será rojo, se le concatena la palabra red

al texto del gráfico en el espacio para cambiar el color del nodo. A continuación se muestra un ejemplo de como quedaría el texto que genera la gráfica:

```
digraph G {
start -> Asia;
Asia -> Japon;
Asia -> China;
Asia -> Korea;
start -> America;
America -> Canada;
America -> Guatemala;
America -> Chile;
start[shape =Mdiamond label="Grafico"];
Asia[shape=record label="{Asia|71}"style=filled fillcolor=
orange];
America[shape=record label="{America|40}"style=filled fillcolor=
green];
Japon[shape=record label="{Japon|80}"style=filled fillcolor=
red];
China[shape=record label="{China|95}"style=filled fillcolor=
red];
Korea[shape=record label="{Korea|40}"style=filled fillcolor=
green];
Canada[shape=record label="{Canada|20}"style=filled fillcolor=
blue];
Guatemala[shape=record label="{Guatemala|40}"style=filled fillcolor=
green];
Chile[shape=record label="{Chile|60}"style=filled fillcolor=
yellow];
}
```

El gráfico, por ejemplo, queda de la siguiente forma:

Para el país seleccionado, primeramente, se utilizó un método de ordenamiento (Bubble sort), siendo el más práctico y efectivo. Se guardó en una matriz dinámica la saturación de cada país, luego se ordenó utilizando el bubble sort. Una vez ordenado, como la saturación se encuentra guardad en cada objeto país. Se toma la saturación de cada país y se compara con el de la posición 0 del arreglo ordenado, de ser el mismo se despliega en pantalla el nombre del país, su población y su ruta. Este mismo método se utilizó para comparar el segundo criterio, que consiste en comparar la saturación del país y aparte la saturación del continente en el que se encuentra, tomando el de menor valor.